SPECIFICATION

At page 1 of the present specification, please rewrite the title as –**PROCESSOR AND METHOD OF EXECUTING A LOAD INSTRUCTION THAT DYNAMICALLY BIFURCATE A LOAD INSTRUCTION INTO SEPARATELY EXECUTABLE PREFETCH AND REGISTER OPERATIONS—.**

Please rewrite page 4, paragraph 3 as follows:

Figure 2A and **2B** illustrate two alternative embodiments of the translation of UISA load instructions into separately executable PREFETCH and REGISTER operations in accordance with the present invention; [[and]]

Please rewrite page 4, paragraph 4 as follows:

Figure 3 is an exemplary load data queue that may be utilized to temporarily buffer load data in accordance with the present invention[[.]]; and

Please insert page 4, paragraph 5 as follows:

Figure 4 is a high level logical flowchart of the processing of a load instruction in accordance with at least one embodiment of the present invention.

Please insert the following paragraphs at page 14, line 11:

Figure 4 is a high level logical flowchart of the above-described process by which a processor processes a UISA load instruction as separately executable IISA PREFETCH and REGISTER instructions. As shown, the process begins at block **100** and thereafter proceeds to block **102**, which depicts ITU **18**/ISU **120** translating UISA load instruction into two, separately executable IISA PRE and REG (or LOAD) instructions. Next, as shown in cycle 1 of Table I and at block 110 of **Figure 4**, ISU **20** detects the PRE instruction and dispatches the PRE instruction out-of-order with respect to a preceding instruction to LSU **26**.

As depicted at block **112**, during cycle 2, LSU **26** executes the PRE instruction to calculate a speculative target address (EA) of the data to be loaded without regard as to whether the contents of the register(s) utilized to compute the target address of the load data may be updated by an instruction executed between the PRE instruction and the associated REG instruction. This target address is then utilized at block **114** to speculatively prefetch data associated with the speculative target address into cache (e.g., L1 data cache **15**). Processing of the PRE instruction thereafter terminates at block **124**.

Referring now to block **120**, during cycle 4, ISU **20** decodes and dispatches the REG instruction to LSU **26**. Next, as illustrated at block **122**, the REG instruction is executed by LSU **26** in cycle 5 and thereafter is completed. As noted above, to execute the REG instruction, LSU **26** computes the EA of the load data and supplies the EA to cache hierarchy **16**, which translates the EA to a real address and determines whether the load data associated with that real address is resident in L1 data cache **15**. Because of the earlier speculative execution of the PRE instruction, in most cases the load data is resident in L1 data cache **15**, and the REG instruction can both execute and load data into one of register files **32** or **34** in the minimum data access latency permitted by cache hierarchy **16**. Following block **122**, the process ends at block **124**.